**Noname manuscript No.**
(will be inserted by the editor)

# Model vs System Level Testing of Autonomous Driving Systems: A Replication and Extension Study

**Andrea Stocco · Brian Pulfer · Paolo Tonella**

Received: date / Accepted: date

**Abstract** Offline model-level testing of autonomous driving software is much cheaper, faster, and diversified than in-field, online system-level testing. Hence, researchers have compared empirically model-level vs system-level testing using driving simulators. They reported the general usefulness of simulators at reproducing the same conditions experienced in-field, but also some inadequacy of model-level testing at exposing failures that are observable only in online mode.

In this work, we replicate the reference study on model vs system-level testing of autonomous vehicles while acknowledging several assumptions that we had reconsidered. These assumptions are related to several threats to validity affecting the original study that motivated additional analysis and the development of techniques to mitigate them. Moreover, we also extend the replicated study by evaluating the original findings when considering a physical, radio-controlled autonomous vehicle.

Our results show that simulator-based testing of autonomous driving systems yields predictions that are close to the ones of real-world datasets when using neural-based translation to mitigate the reality gap induced by the simulation platform. On the other hand, model-level testing failures are in line with those experienced at the system level, both in simulated and physical environments, when considering the pre-failure site, similar-looking images, and accurate labels.

**Keywords:** Autonomous Driving, Model Testing, System Testing, DNN Testing, Deep Neural Networks

A. Stocco and P. Tonella
Università della Svizzera italiana (USI), Via Buffi, 13 – Lugano, Switzerland
tel +41 58 666 40 00, fax +41 58 666 46 47
E-mail: {andrea.stocco,paolo.tonella}@usi.ch
B. Pulfer
Université de Genève, 24 rue du Général-Dufour – 1211 Genève 4
tel +41 (0)22 379 71 11 fax +41 (0)22 379 11 34
E-mail: brian.pulfer@unige.ch

## 1 Introduction

Self-driving cars (SDCs) are autonomous cyber-physical systems capable of sensing the environment and moving safely within well-established and pre-defined scenarios. SDCs deployed on public roads embed a large amount of software (estimated as +100 million lines of code [2]), among which advanced Deep Neural Networks (DNNs) used as perception units to process digital images representing driving scenes and predict the driving control parameters of the vehicle [13, 43, 59, 72]. This complexity makes half of the entire development budget attributed only to testing [27]. Due to the virtually unlimited number of driving scenarios that DNN-based SDCs should support, this cost is expected to grow when testing new vehicle models and versions.

In the literature, two main approaches are used to test DNNs that perform driving tasks. The first approach is *model-level testing* — also referred to as offline testing, whereas the second approach is called *system-level testing*, or online testing [18, 24]. In model-level testing, the DNN is used as an independent unit of computation, and it is fed with a set of labeled driving images retrieved from the real-world, or artificially generated, e.g., by a driving simulator. The DNN predicts values that are compared to the ground truth labels, which serve as an oracle. The difference between the prediction and the ground truth label is called error, and a test is considered failed (model-level failure) when such an error is higher than some predefined threshold.

Differently, in system-level testing, the DNN is embedded within the operational ecosystem in which it is designed to operate, such as a physical vehicle or a driving simulator. While the DNN still processes a stream of (unlabeled) driving images captured by the onboard camera, its predictions have an immediate effect on the overall system behavior, as each prediction and driving decision influence future driving decisions. Thus, the individual DNN's prediction errors become not only less meaningful but also uncomputable, because it is not possible to associate a ground truth label to incoming data. As such, failing tests are characterized in terms of the misbehavior of the whole system in response to the DNN's predictions. A system-level failure is experienced when the system no longer fulfills its safety requirements, such as excessive departure from the driving lane.

While both testing approaches are adopted for ensuring the reliability of DNN-based SDCs, traditionally model-level testing has been more prevalent because of the availability of open-source driving datasets that can be readily used, such as Udacity's [60] or Waymo's [66]. Moreover, it does not necessitate the effort of embedding the DNN within a driving simulator (or a real vehicle), a time-consuming and daunting process.

Researchers have compared the two testing levels and highlighted their differences [18, 24, 25]. In particular, the paper by Ul Haq et al. [24] compares model/system failures within the PreScan simulator. In their work, the authors first assess that virtual tests can be considered an adequate proxy for on-road testing, as their reproduction of the real-life driving conditions of the Udacity dataset [60] within PreScan yields similar external behavior of the DNNs (i.e., steering angle prediction errors) as the real-world. Second, they evaluate two deep neural networks vs an autopilot with global knowledge at driving different scenarios to assess the level of agreement between model and system-level failures. Their results show high disagreement between the failures detected by the two testing

levels. More specifically, the paper reports a large number of false negatives, i.e., failing system-level scenarios in which the individual model-level prediction errors were found to be acceptable. The authors explain that these failures are caused by an accumulation of errors during online driving that is not observable during model-level testing.

We identified three main threats to validity in the work by Ul Haq et al. [24]. First, when comparing real-world and simulated behaviors, driving scenarios are matched by the similarity of the predicted steering angles, not by comparing the images used by the DNNs to make their predictions. Second, when comparing model and system failures, the matched driving scenarios are likely to contain remarkably different input images because the different technologies involved, i.e., DNN vs autopilot with global knowledge, may have different driving behaviours and hence may follow different trajectories. As a consequence, the ground truth provided by the autopilot is a quite imprecise proxy for the *real* ground truth that the DNN should target. Third, when comparing model and system failures, error metrics are averaged on the entire scenario (deep neural network's and autopilot's), instead of considering the behavior of the DNN/autopilot in the immediate proximity of the system failure, when model-level errors are more likely to occur.

In this paper, we replicate the study by Ul Haq et al. [24], improving the experimental setting of the original paper to address the identified threats to the validity. More in detail: (1) to improve the association between simulated and real-world images, on which to compare the DNN behavior, we take advantage of *neural translation techniques*. (2) To mitigate the false negatives possibly due to error metric averaging over the entire test scenario, we focus the comparison only on a sequence of online images that precede the failure (*pre-failure window*). (3) To retrieve accurate labels, we perform *visual similarity matching* between corresponding pairs of online/offline images. Moreover, our work doubles the comparison between model- and system-level failures as we consider both a DNN operating in a simulator and a DNN driving a radio-controlled (RC) physical self-driving car. Our extension to the physical dimension is a novel contribution to the literature.

We reproduced the same results and obtained similar conclusions as the replicated paper [24] on the usefulness of the simulator to produce comparable offline prediction errors as in the real world. Thanks to our improvements in the matching of images between simulated and real-world, we obtained error distributions that are, in many cases, statistically indistinguishable. In the original study, the compared distributions were significantly different, with a large effect size, despite the small prediction error differences. Hence, our study provides stronger statistical support to the findings of the original paper.

Our experimental results also show that violations of the offline oracle (i.e., the occurrence of a high model-level prediction error) have a small number of system-level false positives (i.e., non-corresponding system-level failures) and a small number of false negatives (i.e., system-level failures missed by the model level oracle), suggesting a high agreement between model vs system-level testing. These findings are in contrast with one of the findings of the replicated paper by Ul Haq et al. [24]: "Offline testing is more optimistic than online testing because the accumulation of errors is not observed in offline testing." In fact, by considering the pre-failure window and by accurately matching pairs of images using visual similarity, we were able to observe an accumulation of offline errors in most image sequences leading to an online failure.

The paper is structured as follows. Section 2 reports background information. Section 3 describes the replicated study and the threats to validity. Section 4 illustrates the empirical study in which we report our mitigation strategies to the threats to validity affecting the original paper, our extension to a physical SDC, and detailed statistical analysis. In Section 5 we provide a qualitative analysis of our results. Section 6 gives an overview of the related work, while Section 7 concludes the paper.

## 2 Preliminaries

### 2.1 Autonomous Driving Software

Most existing SDCs are vehicles equipped with specific sensors (such as cameras, LIDAR and GPS) used to perform different prediction and control tasks, such as lane-keeping, object avoidance, and path planning, to name a few. SDCs use the information collected by the sensors during a supervised data collection process to train DNNs at predicting control values that are sent to the car's actuators to perform the actual maneuvers (e.g., steering).

In this work, we study SDC models that perform imitation learning for *lane-keeping*, i.e., a supervised learning task in which the DNN learns how to keep the position of the vehicle in lane, by predicting the steering angle control from a dataset of driving scenes labeled during a driving session with a human driver.

Our focus is the comparison between model (offline) and system (online) testing on a closed-loop track, a widely adopted industrial practice that precedes on-road testing on public roads [30, 58, 11, 16, 65, 67].

### 2.2 Model-level Testing

After the training process, DNN models are tested by measuring evaluation metrics on test datasets [45] such as accuracy or mean squared error (see Fig. 1).

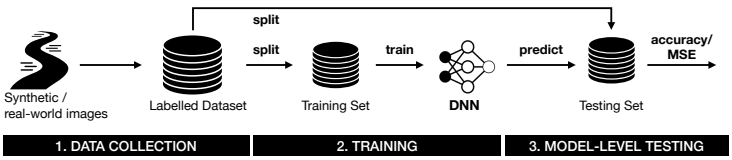


Fig. 1: Model-level testing of autonomous vehicles.

We refer to this modality of testing as *model-level testing* [45], because the model is tested as a standalone component, evaluating only the predictions the DNN makes on individual images. This level of testing is comparable to unit testing for traditional software and can be useful to reveal faults in the training process (e.g., suboptimal learning rate), or in the quality of the data used for training (e.g., training data imbalance) [28].

For lane-keeping DNNs, error metrics measure the difference between the DNN's predictions and the ground truth values, e.g., by computing the mean squared error or the mean absolute error. Let us take a driving sequence $d$ composed of $n$ images. The mean absolute error (MAE) is given by:

$$MAE(d) = \frac{1}{n}\sum_{i=1}^{n}|y_i - f(x_i)|$$

where $x_i$ denotes the $i$th image in the driving sequence $d$, $f(x_i)$ the output of the DNN, and $y_i$ the ground truth value. A failure of the DNN, in model-level (offline) testing, is defined by a MAE above a certain threshold $\epsilon$.

$$Failure(d) = \begin{cases} True, & \text{if } MAE(d) \geq \epsilon \\ False, & \text{otherwise} \end{cases}$$

Thresholds confine the maximum tolerable prediction error within validity ranges that are defined based on the domain knowledge of test engineers for the possible classes of images (i.e., straight roads, bends, weather conditions).

The main advantages of model-level testing rely upon its simplicity and low requirements needed to execute it. The disadvantages consist in being *stateless*, which makes it ineffective at revealing faults occurring when the DNN is deployed in production. Indeed, when the DNN model is tested within a vehicle, the whole system can compensate for some high DNN inaccuracies or, on the contrary, it might be affected by the accumulation of small DNN inaccuracies over time.

### 2.3 System-level Testing

To overcome the limitations of model-level testing, *system-level testing* requires embedding the DNN within a SDC to test the whole decision-making process. System-level testing is *stateful* as it allows to observe the effects that the predictions made by the DNN have on the behavior of the entire system (Fig. 2).

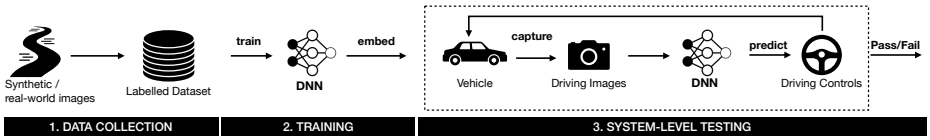


Fig. 2: System-level testing of autonomous vehicles.

With system-level testing, it is possible to gather concrete values of system quality metrics [31], such as the speed or the position of the vehicle. Thus, a system-level failure (or online failure) is characterized as one of the system quality metrics being higher than a threshold determined by the environment (e.g., the road width), by the regulations in which the system operates (e.g., the speed limit), or by safety requirements (e.g., the vehicle drives off-road or causes harm to other vehicles, to the environment, or people).

Fig. 3: Example of simulator-generated driving image from PreScan corresponding to a real-world image [24]

The main advantages of system-level testing consist in the exposure of actual requirement violations, as failures are associated with the external behaviour of the software in response to the DNN predictions. Extensive system-level testing is pivotal when the final goal is the deployment of the SDC on public roads, which is subject to strict regulations [30,58]. However, the main disadvantage of system-level testing consists in its high execution cost, as it necessitates embedding the DNN within a driving simulator or a real vehicle, in addition to the time required to run extensive test-driving sessions (both virtual or in the field).

## 3 Replicated Study

This paper is a replication and extension of the work by Ul Haq et al. [24] presented at the International Conference on Software Testing, Validation and Verification (ICST) in 2020. An extended version of the paper has been published in the Empirical Software Engineering journal [25]. In this paper, we consider the conference version of the work.

The work by Ul Haq et al. [24] (replicated paper, hereafter) aims to test a lane-keeping DNN trained with real-world data of the Udacity dataset [60], both at model and system level. The authors consider simulation-based testing within the driving simulator PreScan. The capability of PreScan to reproduce the real-world conditions of the Udacity dataset [60] is assessed in the first research question:

**$RQ_0$:** *Can we use simulator-generated data as a reliable alternative source to real-world data?*

The authors performed a visual assessment of the Udacity dataset [60] to infer the road characteristics and environmental conditions. Then, they instrumented PreScan to generate driving scenarios that are in line with the retrieved characteristics. Finally, they selected sequences of simulator-generated scenarios that are similar to the real-world data. In their work, the similarity was measured by means of heuristics that consider only the DNN's prediction, i.e., the steering angle predicted by the DNN either on simulated or on real images. A pair of real and simulated scenarios is considered matched if the scenarios have the same length and the average difference between pairs of predicted steering angles is lower than a threshold $\epsilon$ (set to 2.5°). The authors report that the large majority of computer-generated "scenarios (92/100) could match subsequences of the Udacity real-life test dataset".

Fig. 3 shows an example of image match used in the original paper, from which it is evident that visual similarity of the matched images is low (e.g., shadow or cloudy sky are not represented in the real-world image). Despite the generally low resemblance of simulator-generated sequences, empirical results show that the DNN's "prediction error differences between simulator-generated datasets and real-life datasets are less than 0.1, on average". Thus, the authors "conclude that we can use simulator generated datasets as a reliable alternative to real-world datasets for testing DNNs".

**Threat $RQ_0$-T1: poor visual similarity between real and simulated road images.** The authors recognize the poor visual similarity between real-world and simulator-generated data in the threats to validity. One goal of our replication is to address the main threat to validity affecting the original authors' findings for $RQ_0$: the poor visual similarity between real and simulated road images. Indeed, when poor visual similarity affects the comparison, the compared DNNs will learn features that are quite different between real-world and simulated images as DNNs used for driving make use of convolutional layers as feature extractors. For instance, considering Fig. 3, a DNN will extract geometrical features both from the double solid yellow lines on the left as well as the single solid white line on the right for the real-world image. The simulated image does not represent lane conditions, as the right lane is occluded by a shadow, not present in the real-world, and the left lane is a single broken white line. In our paper, we address such a threat by adopting an automated approach based on neural image translation (see Section 4.4.4).

The second and main research question of the replicated paper focuses on the comparison between model- and system-level testing on the simulation platform:

**$RQ_1$:** *How do offline and online testing results differ and complement each other?*

The authors extended the set of conditions that can be generated by PreScan, including weather effects to create unseen scenarios that could expose failures of the DNNs under test at the system level. Then, they generated 50 random scenarios and used the autopilot module of PreScan to generate a ground truth driving trajectory (i.e., sequence of steering angles). We contacted the first author and asked for clarifications about the computation of the ground truth steering angles; the response was quick and detailed.

They executed two pre-trained DNNs models from the literature, Autumn [13] and Chauffeur [57], on the same 50 random scenarios, to collect predicted steering angles, as well as the Maximum Distance from the Center of Lane (MDCL). For a large set of the generated scenarios (87%), system-level failures occur based on the observed MDCL, which was above a threshold of 0.7 (corresponding to approximately 1.5 meter). Then, for each generated scenario (both failing and non-failing), they computed MAE, as the mean difference between the sequence of predicted steering angles and the sequence of ground truth steering angles from the autopilot's driving trajectory. A MAE above 0.1 (2.5°) was regarded as a model-level failure. Scenarios in which the MAE/MDCL are both above or both below their respective thresholds are said to be *in agreement*, otherwise, they are regarded as being *in disagreement*. The authors found large disagreement in the case of system-level failures, in contrast to the high agreement for the non-failing conditions, or for model-level failures.

We have identified two main threats to the validity of the original experimental design that we wish to address in this work. We use a graphical support to illustrate our hypothesis. Fig. 4 (left) shows a typical generated test scenario with a road

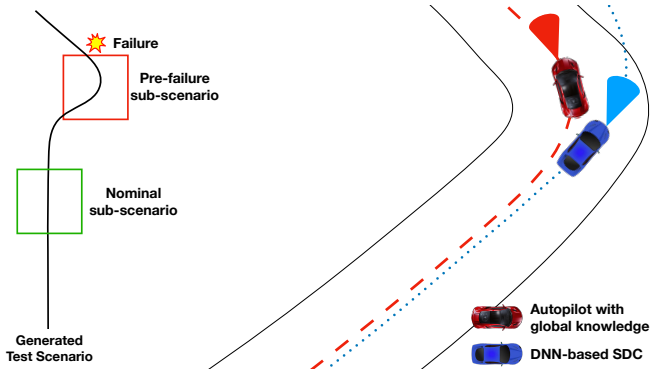


Fig. 4: Scenario-level matching may cause different sets of images to be compared as it typically combines both nominal and failing sub-scenarios.

characterized by an initial road segment, a curve on the right followed by a curve on the left. Fig. 4 (right) shows the trajectories taken by the autopilot with global knowledge, which is used as a reference for the ground truth steering angles, and by the DNN-based SDC under test.

**Threat $RQ_1$-T1: different driving conditions experienced by autopilot vs DNN.** Although the driving scenario, among the 50 that have been generated, is kept the same when autopilot or DNN are driving, the sequence of images captured and processed by the autopilot is likely to be quite different from the sequence of images captured and processed by the DNN. Indeed, from the replicated paper, it is reported that, for 87% of the cases, the outcome of the simulation diverges between the autopilot and the DNN, as the autopilot is expected to exhibit no failures, thanks to its access to global knowledge. Hence, the labels by the autopilot are not reliable because they have been obtained on a set of images that are different from the ones experienced by the DNN, especially during near-failing sub-sequences of the test scenarios (see Fig. 4 right). Thus, for the computation of the MAE, more reliable labels should be used for assessing offline DNN failures.

In our work, we address $RQ_1$-T1 as follows: we perform a search using a state-of-the-art visual matching algorithm, Structural Similarity Index (SSIM) [64], to match each individual driving frame observed by the DNN within the pre-failure sequence with the closest labeled driving frame available in the training set. We use the label of such a matching frame from the training set as the ground truth steering angle for the calculation of the MAE.

**Threat $RQ_1$-T2: MAE computed on entire driving scenario.** In the results tables of the original paper [24], the case "MAE $< 0.1$ and MDCL $\geq 0.7$" is the second most prevalent, while it becomes the most prevalent in the journal extension of the replicated paper [25]. This can be interpreted as most DNN predictions being correct even when the vehicle is departing from the road. The authors motivate this as "Offline testing is more optimistic than online testing because the accumulation of errors (eventually causing a critical lane departure) is not observed in offline testing." [24]. While the motivation provided by the authors is intuitive, we hypothesize another explanation for these results.

Since it is not possible to retrieve the ground truth for each individual image observed and processed by the DNN, because the autopilot might have never seen exactly the same image, the authors rely on a coarse-grained matching, performed by considering the entire sequence of steering angles in the same scenario driven by both autopilot and DNN (threat $RQ_1$-T1). Correspondingly, MAE values are computed as the average over all the images observed in an entire driving scenario and a model-level failure occurs only when such average is above the threshold 0.1. This has the disadvantage to include in the comparison also many images in which the car is not yet deviating from the lane, along with the corresponding (presumably) correct steering angles. While in this case the frames in which the DNN-based vehicle is not yet deviating are likely to be more similar to the reference autopilot's frames, *both nominal driving and pre-failure driving sub-scenarios are considered within the same sequence-level comparison* (see Fig. 4). If the nominal sub-scenario dominates the pre-failure sub-scenario, the MAE would result below the threshold even for failing scenarios.

To address $RQ_1$-T2, we adopt the following mitigations: we restrict the computation of MAE to a pre-failure sub-scenario, which occurs before off-road driving. While a precise identification of the pre-failure window may be challenging in most real-world settings, our experimental framework allows us to have full control. Specifically, our driving simulator logs each frame with the position of the car on the track. Concerning real-world data, the position of the car is estimated by a DNN trained for that purpose [51]. Thus, we can identify precisely the first driving frame in which the car departs from the drivable road section. Based on this precise definition of system-level failure, we isolate a pre-failure window of driving frames, as well as the associated predictions, that precede each system-level failure. Then, we utilize only the pre-failure window when performing the comparison between model vs system-level testing oracle violations.

## 4 Empirical Study

In our empirical study, we compare model- and system-level testing of both physical and virtual SDCs. The goal of the study is to assess whether the results from the replicated study [24] hold when improving the experimental setting and when considering the physical platform Donkey Car [21] in addition to its digital twin.

### 4.1 Research Questions

We consider the same research questions of the replicated study [24], which have been briefly presented in Section 3:

**$RQ_0$:** *Can we use simulator-generated data as a reliable alternative source to real-world data?*

In the replicated study, the authors rely on a stationary dataset of real-world images, for which driving quality metrics (i.e, MDCL) are not available. Moreover, the authors could not reproduce the same driving conditions on a real-world vehicle, because they relied on Udacity's pre-collected images. For this reason, they first investigate whether they could rely on the virtual images from a simulator to test a DNN trained with real-world data.

Differently, in our work, we consider on-road system-level testing with a physical vehicle (*hardware-in-the-loop*), instead of relying on the stationary dataset of images provided by Udacity [60]. This obviates the need of demonstrating the representativeness of simulator-generated data as we can directly measure, or estimate, the in-field quality metrics for system-level testing. Nevertheless, we study $RQ_0$ by addressing **$RQ_0$-T1** through unsupervised image neural translation techniques for the reconstruction of real-world scenes within a simulator.

**$RQ_1$:** *How do offline and online testing results differ and complement each other?*

$RQ_1$ is the main research question of the paper. We take advantage of the Donkey Car framework to compare the failure profiles observed in the virtual vs the physical world, both at the model- and system-level, by addressing **$RQ_1$-T1** and **$RQ_1$-T2** through pre-failure window selection and visual similarity matching.

### 4.2 Self-Driving Car Models

We test the same two DNN-based SDCs of the replicated study [24]: Autumn [56] and Chauffeur [57]. These publicly available SDC models scored high rankings in the Udacity challenge and they have been used as experimental subjects in several testing works [31,43,52,53,54,59,72]. Autumn consists of three convolutional layers, followed by five fully-connected layers [56]. Chauffeur uses six convolutional layers to extract the features of input images, two-dimensional dropout layers, and a fully connected layer [57].

### 4.3 The Platform

While full-scale testing of SDC is still impractical for most academic settings as it presents severe time, space, and cost constraints [14,62], small-scale vehicles represent an interesting alternative. Frameworks such as Donkey Car [21] or AWS DeepRacer [6] are derived from remote-controlled (RC) cars and provide an electrical engine and a battery as a main power unit. Although these are small-scale vehicles, they reach considerably high speeds and accelerations for their size [10]. RC cars are adopted at the early stages of testing autonomous driving algorithms as they retain relevant photorealistic conditions of the driving environments which are experienced also by full-scale cars [62]. These platforms are increasingly used by researchers who want to experiment their solutions on real vehicles for the purpose of testing newly developed autonomous driving software [7,10,32,39,47,63,73,75].

In our study, we adopt the Donkey Car™ open-source framework [21]. Donkey Car includes an HSP 94186 Brushed RC car with self-driving capabilities, a Python framework supporting training and testing of SDCs that perform lane-keeping, and a simulator developed with Unity [61], a popular cross-platform game engine, in which the real-world DonkeyCar's actuators are modeled with high fidelity. Donkey Car is one of the reference platforms for studies comparing the autonomous driving testing of small-scale SDCs [63], because of its open-source nature. In our study, we leverage the Donkey Car framework to perform model-level vs system-level testing of SDCs, both in the virtual and in the real world, with the latter being a totally novel contribution of this study.

#### *4.3.1 Testing Tracks*

Our testing track is an 11m long track, printed on a mat of size 3.0m × 4.54m. The road section is 52 cm wide. Clockwise, the track features three curves on the right and one on the left. The Donkey Car simulator features a scene that resembles our real-world track in terms of the road's shape, colors, and proportions [51].

In our setting, the car follows the middle line on a two-lane road (as if it were a single-lane, one-way road) and moves only forward. We use the lateral position, or cross-track error (XTE), to assess the lane-keeping capability of SDC models. XTE measures the distance from the center of the car to the center of the road [52]. The MDCL metric used in the replicated paper measures the distance of the center of the car from the center of the lane of a two-lane, two-way road (instead of the center of the road). Hence, we consider a thresholded XTE (i.e., a Maximum XTE) as comparable to MDCL.

### 4.4 $RQ_0$: Procedure and Results

#### *4.4.1 SDCs Data Collection*

For each testing environment (virtual and physical world), we collect two training sets by manually driving on both the virtual and physical tracks, incentivizing the vehicle to stay close to the centerline of the track. We followed the guidelines by Kramer et al. [55] for generating driving sequences for the DNN. In particular, we followed the suggestion to generate both nominal and near-failing driving sequences [13,55]. Nominal driving sequences are those that incentivize the vehicle to stay on track. Near-failing driving sequences are used to teach the DNN how to recover the vehicle back to track. Both driving styles are needed for the training of a robust lane-keeping DNN [13,55], in order to make the DNN able to cope with different driving conditions possibly occurring in the same track sector. We kept a constant throttle value of 0.3, resulting in a maximum driving speed of 3.1 mph (5 km/h, or 1.40 m/s) during data collection. Images are acquired from the front-facing camera at 21 frames per second (FPS), labeled with the ground truth steering angle of the human driver.

Fig. 5 shows the distributions of steering angles of our training sets for both testing environments. Average steering angles are 0.314±0.461 for simulated and 0.316±0.431 for real-world training sets, respectively.

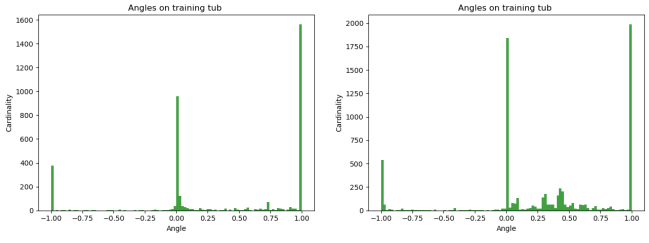


Fig. 5: Steering angles distributions for the virtual (left) and real-world (right).

#### *4.4.2 SDCs Model Setup & Training*

For each DNN (Autumn, Chauffeur), we trained an individual SDC model on each training set (virtual and real-world), for a total of four models. Following the guidelines by Bojarski et al. [13] for the hyper-parameters, the number of epochs was set to 500, with a batch size of 64 and a learning rate of 0.0001. We used early stopping with a patience value of 30 and a minimum loss change of 0.0005 on the validation set. The DNNs use the Adam optimizer to minimize the MSE between the predicted steering angles and the ground truth values. As common practices require, we cropped the images to $140 \times 320$ by removing 100 pixels from the top, which allows the DNN to focus on the part of the image relevant for lane-keeping. We used data augmentation (e.g., translation, brightness) to increase the diversity in the training data.

#### *4.4.3 SDCs Sanity Check*

After training, we assessed that the four trained models are robust enough to be considered in the subsequent testing phase. We let them drive in their corresponding testing tracks multiple times and observed that they can drive without crashing or going off-road. For the physical vehicle, we also controlled the discharge of the Donkey Car's battery and we recharged the battery if the voltage was found to jeopardize the overall quality of driving.

#### *4.4.4 Mitigating $RQ_0$-T1 with CycleGAN*

The distribution of real-world images can be different from the distribution of simulator-generated images [4] (threat **$RQ_0$-T1**), undermining the validity of our study. To mitigate **$RQ_0$-T1**, we use a generative adversarial network (GAN) called CycleGAN [76] to generate real-world driving images from the corresponding simulated ones. CycleGAN is a cycle-consistent adversarial generative network that performs an unsupervised and unpaired image-to-image translation. The two datasets of images do not need to be paired, yet they should represent analogous driving data images (respectively, images from the simulated track and images from the real-world track). Starting from two sets of images with analogous latent features, CycleGAN learns two image-to-image encoder-decoder functions that share the latent space, so that, given an image from one domain (e.g., a virtual driving scene), it is possible to generate not only a similar image in the same domain but also the corresponding image in the other domain (e.g., a real-world driving scene). Fig. 6 shows an example of neural translation in which we use a CycleGAN model to convert a real-world image (left) into a virtual image (right).

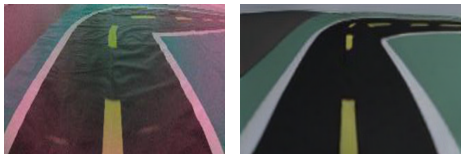

Fig. 6: Neural-generated driving image corresponding to a real-world image.

#### *4.4.5 Comparing real-world vs virtual driving*

From the replication package provided by the authors [24], we were able to obtain the predictions, ground truth, and MAE values of 31/92 sequences. Correspondingly, we collected 31 real-world scenarios by manual driving: these scenarios are labeled with humanly produced ground truth steering angle values, for a total of 7,906 real-world images. We used CycleGAN to translate them into 31 corresponding virtual scenarios. Usage of CycleGAN ensures a high visual similarity between the real-world image and its translation into a simulated image, which was not the case of the replicated study (Fig. 3).

We executed our SDC models in offline mode, and we compared the obtained steering angle predictions with the ground truth steering angles to obtain per-frame absolute errors. We performed a statistical comparison between the prediction error distribution obtained for real-world scenarios and the prediction error distribution for the simulated scenarios. We assess the statistical significance of the differences between real-world and simulator errors using the non-parametric Mann-Whitney U test [68] (with $\alpha = 0.05$), the magnitude of the differences using the Cohen's $d$ effect size [19], and the statistical power with a Monte Carlo power analysis [15] with 80% power target as our data is not normally distributed.

#### *4.4.6 $RQ_0$: Results*

Table 1 reports the results about the prediction error differences between simulator-generated data and real-world data. For each SDC model, we report the input type used during training (Train) and testing (Test), the average MAE difference between real-world and reconstructed virtual scenarios, and the percentage of simulations for which the MAE difference was below the threshold $\epsilon = 0.1$ (2.5°) used in the replicated study.

The first observation is that all models attain an average MAE difference $< 0.1$, which is consistent with the results reported in the replicated study [24]. This happens in our improved experimental setting, which mitigates **$RQ_0$-T1** with neural translation, as well as in our replication of the imprecise image matching described in the replicated study [24] (last two rows of Table 1). As expected, the MAE difference is higher when offline testing a DNN-based SDC trained on real-world data onto the simulation platform.

Table 1: $RQ_0$: Prediction error differences between simulator and real-world data.

| | Input type | | Avg MAE | Simulations $< \epsilon$ |
|---|---|---|---|---|
| | Train | Test | Diff | % |
| **Our Study** | | | | |
| Autumn | Virtual | Virtual | 0.01 | 68 |
| Chauffeur | Virtual | Virtual | 0.02 | 100 |
| Autumn | Real-world | Virtual | 0.06 | 39 |
| Chauffeur | Real-world | Virtual | 0.02 | 90 |
| **Replication of** [24] | | | | |
| Autumn | Real-world | Virtual | 0.08 | 74 |
| Chauffeur | Real-world | Virtual | 0.06 | 84 |

Table 2: $RQ_0$: Statistical analysis of the prediction errors between real-world driving sequences and their virtual reconstruction.

| | negligible | | | small | | | medium | | | large | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $p < .05$ | $p \geq .05$ | | $p < .05$ | $p \geq .05$ | | $p < .05$ | $p \geq .05$ | | $p < .05$ | $p \geq .05$ | |
| | | pow | | | pow | | | pow | | | pow | |
| | | $< .8$ | $\geq .8$ | | $< .8$ | $\geq .8$ | | $< .8$ | $\geq .8$ | | $< .8$ | $\geq .8$ |
| **Our Study** | | | | | | | | | | | | |
| Autumn | 1 | 16 | 5 | 3 | 0 | 5 | 1 | 0 | 0 | 0 | 0 | 0 |
| Chauffeur | 10 | 10 | 7 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| Autumn | 1 | 4 | 7 | 13 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 |
| Chauffeur | 5 | 24 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Replication of [24]** | | | | | | | | | | | | |
| Autumn | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 30 | 0 | 0 |
| Chauffeur | 3 | 3 | 2 | 5 | 0 | 2 | 3 | 0 | 0 | 13 | 0 | 0 |

Concerning the statistical analysis of prediction error distributions (not done in the original paper [24]), Table 2 classifies the simulations of Table 1 according to whether the distribution of prediction errors was different with statistical significance ($p$-value $< 0.05$) and those for which it was the same for real-world and for translated simulator images (i.e., $p$-value $\geq 0.05$, divided by low/high statistical power). Results are further divided by effect size (negligible, small, medium, large). We can notice from Table 2 that in our replication of the original study, the majority of the simulations have prediction errors distributed quite differently from those obtained from real-world, Udacity images, with statistically significant differences and large effect size. Only 3% for Autumn and 23% simulations for Chauffeur have a negligible/small effect size, and only 13% are supported by power analysis. Actually, for Autumn, 30/31 simulations (97%) and 24/31 (84%) simulations in the case of Chauffeur have a large effect size.

In our improved experimental setting, which takes advantage of neural translation, prediction errors tend to be closer between simulated and real images, in most cases. When DNN models are trained on simulated images, the two prediction error distributions (obtained on simulated vs real images) exhibit negligible differences in 84% (Autumn) and 68% (Chauffeur) of the cases. When DNN models are trained on real-world images, prediction errors have negligible differences in 55% (Autumn) and 77% (Chauffeur) of the cases (with a negligible effect size, power analysis requires a huge number of samples to reach the threshold of 0.8).

**$RQ_0$**: *The prediction error differences between simulator-generated and real-life datasets are less than 0.1, on average, for both Autumn and Chauffeur, confirming previous results. Statistical analysis of the prediction error distributions revealed statistically significant differences with large effect size in the original experimental setting. In our experimental setting, with improved image matching due to neural translation, statistical analysis reports negligible differences between the error distributions, providing a stronger statistical support to the original findings on the usefulness of the simulator to trigger similar DNN behaviors as in the real world.*

## 4.5 $RQ_1$: Procedure and Results

In RQ1 we perform a comparison between model and system-level testing, for the virtual and real-world SDC separately.

### *4.5.1 Generating Test Scenarios*

As our SDC models are constructed to be failure-free in nominal conditions, similarly to the replicated study, we test them by injecting unknown conditions (i.e., conditions different from those in the training set) onto the existing tracks *in real-time during driving*.

We use the black-box image corruptions proposed by Hendrycks et al. [26], commonly used to test DNNs that process imagery data. The paper proposes 18 corruptions belonging to five classes, namely *noise*, *blur*, *weather*, *luminance*, and *resolution reduction*.

We test each SDC using 36 scenarios, of which 18 failure-inducing scenarios and 18 failure-free scenarios. The former were obtained from 72 one-lap simulations (for a total of 4,665 images) by enabling the corruptions for each model Autumn (sim and real) and Chauffeur (sim and real) in their respective environments (virtual and real). All such simulations (4,665 images overall) experienced a system-level failure (Section 2.3) due to image corruption. These failing simulations are used to assess the true alarms reported by model-level testing. The latter were obtained from four one-lap simulations *with no corruption* enabled, one for each model, Autumn (sim and real), and Chauffeur (sim and real) in their respective environments. All such simulations experienced no system-level failures and are used to assess whether false alarms are reported by model-level testing.

### *4.5.2 Mitigating $RQ_1$-T2 with pre-failure window selection*

We recall that in our setting, a system-level failure occurs during off-road driving episodes. The simulator automatically flags the car as off-road if the car's position deviates by more than half of the track's width (i.e., $|XTE| > 2.2$, as $XTE = \pm 2.2$ marks the lane borders, whereas XTE = 0 represents the middle of the lane). In the real world, we use an existing telemetry estimator from the literature [51] to automatically retrieve the XTE value for real-world images. Thus, for both settings, based on our definition of system-level failure, we are able to isolate the pre-failure sub-scenario of driving frames (and predictions) that precedes each system-level failure, which mitigates threat $RQ_1$-T2.

Fig. 7 illustrates an example from our empirical study. In the figure, the Chauffeur model drives on our simulated version of the testing track from right to left. An image corruption of type "fog" is automatically injected onto the original camera frame (corrupted images). Each image is labeled with the steering angle (SA) predicted by Chauffeur and the XTE value. The simulation fails when $|XTE| > 2.2$, which occurs for the leftmost frame of the figure. We refer to this frame as the *first failing image*. Thus, we consider a sequence of images preceding the first failing image as the potential candidate for the root cause of the failure, i.e., the sequence in the vicinity of the failure site in which most wrong predictions are expected to have occurred. We have considered a pre-failure sub-scenario of 3 seconds, corresponding to 63 frames (Section 4.4.1), a reasonable value found

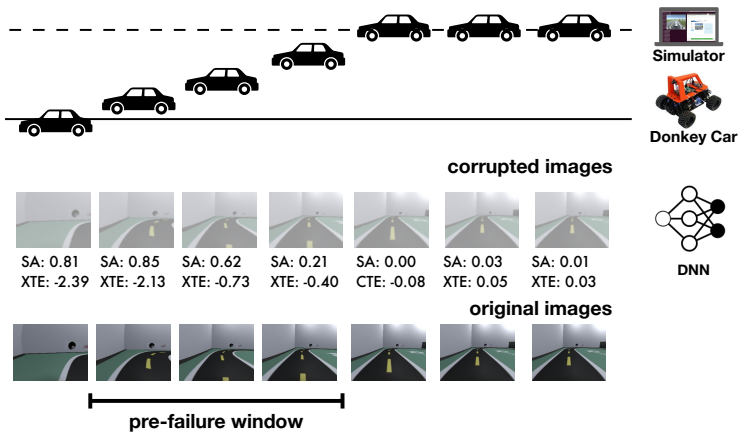


Fig. 7: Pre-failure window selection.

during preliminary experiments, given the relative shortness of our testing track. We refer to such sequence as the *pre-failure window*.

### *4.5.3 Mitigating RQ₁-T1 with visual similarity search*

To mitigate threat $RQ_1$-T1, we perform pre-failure window selection, needed to find the most similar image with a ground truth label, using a visual similarity metric called SSIM [64] (structural similarity index). SSIM simulates the high sensitivity of the human visual system to structural distortions while compensating for non-structural distortions. It is considered a more reliable measure to the per-pixel metrics such as Euclidean distance. SSIM is a floating-point number that ranges from 0 (no similarity) to 1 (perfect match).

We calculate the SSIM score between each image in the pre-failure window and each image in the subset of the training set that is related to the portion of the track in which the pre-failure window occurs. Thus, we select the image with the highest visual similarity (i.e., highest SSIM score) as the reference image to use as the ground truth. To lower the chance of false matches, we make sure to match frames that belong to the same part of the track. Our tracks are divided into five distinct logical sectors. Our simulation platform labels each image with the corresponding track sector. For real-world images, such information is not available, thus we manually assigned each image to the sector they belong to in our real-world closed-loop testing track.

### *4.5.4 Configurations*

For each scenario, we calculated the per-frame absolute error value, both for the virtual and the real-world scenarios. We also replicated in our setting the scenario-level matching of the replicated study, by matching entire sequences instead of individual frames. Since autopilot modules are not available in our framework, we produced two one-lap simulations by manually driving the tracks, both in

Table 3: $RQ_1$: Results of the comparison between model and system level testing.

| | **Visual search on pre-failure window** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | *virtual* | | | | *real-world* | | | |
| | Autumn | | Chauffeur | | Autumn | | Chauffeur | |
| | MAE $<$ 0.1 | MAE $\geq$ 0.1 | MAE $<$ 0.1 | MAE $\geq$ 0.1 | MAE $<$ 0.1 | MAE $\geq$ 0.1 | MAE $<$ 0.1 | MAE $\geq$ 0.1 |
| Nominal | 18 | 0 | 18 | 0 | 14 | 4 | 13 | 5 |
| Pre-failing | 2 | 16 | 0 | 18 | 2 | 16 | 2 | 16 |
| Total | 20 | 16 | 18 | 18 | 16 | 20 | 13 | 23 |

the virtual and real-world, to obtain a reference driving trajectory (ground truth steering angles) that can be used for computing the MAE over the entire scenario.

### *4.5.5 $RQ_1$: Results*

Table 3 reports the results for our proposed matching technique (visual search on pre-failure window) on both environments (simulation and real-world). For each configuration, the table reports the number of cases in which model-level and system-level testing are *in agreement* (i.e., MAE $<$ 0.1 in nominal scenarios, or MAE $\geq$ 0.1 in pre-failing scenarios) and the number of cases in which they are *in disagreement* (i.e., MAE $<$ 0.1 in pre-failing scenarios scenarios, or MAE $\geq$ 0.1 in nominal scenarios).

Overall, our results show a high agreement between model and system-level testing and are not consistent with the results presented by the original authors. We believe this is due to our refined experimental setting. In the simulated environment, the agreement rate is 94% for Autumn and 100% for Chauffeur. The model level oracle exhibits only two false negatives (i.e., missed system-level failures) for Autumn. In the real-world physical environment, the agreement rate is 83% for Autumn and 86% for Chauffeur. The model level oracle exhibits two false negatives for Autumn, as well as four false positives (wrong expectations of system-level failures, due to high model-level MAE). For Chauffeur, model-level testing reports no false negatives, but five false positives.

Table 4: $RQ_1$: Our replication of $RQ_1$ [24].

| | *virtual* | | | |
|---|---|---|---|---|
| | Autumn | | Chauffeur | |
| | MAE $<$ 0.1 | MAE $\geq$ 0.1 | MAE $<$ 0.1 | MAE $\geq$ 0.1 |
| Nominal | 18 | 0 | 18 | 0 |
| Pre-failing | 11 | 7 | 9 | 9 |
| Total | 29 | 7 | 27 | 9 |

Concerning our replication of the results by Ul Haq et al. [24], we were able to reproduce the results of the original paper (i.e. no false positives) using the scenario-level matching proposed by the authors (Table 4). Model vs system-level testing disagreement is 50% for Autumn (100% of false negatives, no false positives) and drops to 22% for Chauffeur (44% of false negatives, no false positives).

**RQ$_1$**: *Model-level (offline) and system-level (online) testing results agree in most cases when using a scenario matching technique based on the pre-failure site, similar-looking images, and accurate labels. On the contrary, the disagreement reported in the replicated study emerges only when using a scenario matching technique based on the entire image sequence and on the auto-pilot ground truth.*

## 4.6 Threats to Validity

### *4.6.1 Internal validity*

One threat to internal validity concerns our custom implementation of the SDCs, with custom training sets. To mitigate this threat, we implemented best practices [13,55] to make sure to train robust SDC models that exhibited no failures in nominal conditions. Another threat is that the pre-failing images may not find a match in the training set if this does not contain diverse trajectories. However, this scenario never occurred in our experiments.

Lastly, the replicated study uses a simulator in which the car drives on a specific lane of a two-lane road whereas in our setting the car follows the middle line on a two-lane road (as if it were a single-lane, one-way road). While the MDCL used in the replicated paper is a measure of distance from the center of the lane (instead of the center of the road), we consider a thresholded XTE (i.e., a Maximum XTE) as comparable to MDCL.

### *4.6.2 External validity*

The use of the Donkey Car framework poses a threat in terms of the generalizability of our results. While Donkey Car has been used in similar studies for DNN testing [39,62,63,75], generalizability to other physical settings is not guaranteed. We considered only one physical track, instead of open-source datasets of labeled driving images. However, this was unavoidable, as we are not aware of ways to reliably import real-world driving data within a simulation platform, or within the Donkey Car.

### *4.6.3 Reproducibility*

We make our data, results, and the Donkey Car simulator available [1]. The techniques and heuristics proposed in this paper do not need necessarily a physical platform and can be applied, for instance, to stationary datasets as Udacity's [60]. For a complete replication of our study, two open-source physical assets are needed, i.e., the Donkey Car and a racing track with the characteristics described in Section 4.3.1.

## 5 Qualitative analysis

The Autumn DNN model exhibited 4 false positives (see Table 3) when driving the physical Donkey Car. Although this is a small number compared to the true positives and true negatives (resp. 16 and 14), we have investigated them qualitatively in-depth, to understand the core reasons behind a high offline prediction error when the car can drive safely in nominal conditions. Fig. 8 reports some meaningful examples from our experiments.

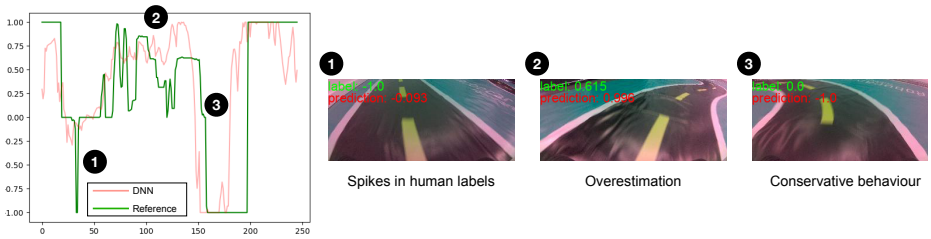


Fig. 8: Examples of wrong predictions causing false positives.

Plot/image ❶ show a case in which, during a straight road segment, the human driver had to correct the trajectory after a bend, due to the high speed of the vehicle (green curve). The SDC model, on the other hand, predicts a steering angle near zero (red curve), which is in line with the average steering angle learned from the training set distribution for straight road segments of this kind (see Fig. 5). Hence, the prediction error is large.

Plot/image ❷ show a case in which the human driver applies a moderate steering angle on the right (≈0.6, or 10°) and travels a right bend at the center of the road. The SDC model, on the other hand, predicts a steering angle near 1.0 (16°) which means full steering on the right, in line with the average steering angle learned from the training set distribution for right road segments, causing a large prediction error.

Finally, plot/image ❸ show a case in which the SDC model predicts left steering angle commands a few frames before the human driver. This anticipating behavior can be explained by the fact that the DNN has learned a conservative behavior towards certain challenging conditions, which deviate substantially from the human ground truth angle.

Overall, we observed a different driving style between humans and DNN, despite the latter is imitating the former. This might generate offline errors that do not correspond to any system-level failure (false positives).

Another finding of this study concerns the generalizability of the results obtained on a simulation platform to the physical environment. We have two main explanations for this: (1) our simulated platform is a digital twin of the physical car (i.e., a faithful virtual replica of the vehicle and its sensors) and (2) we maintained the same experimental setting across virtual and real environments.

## 6 Related Work

### 6.1 Model vs System Testing Comparison

Codevilla et al. [18] investigate the relation between model-level vs system-level testing metrics for SDCs. They use the simulation environment CARLA [22], finding that offline prediction errors are not correlated with driving quality. Moreover, they report that two DNN models with analogous error prediction rates may differ substantially in their driving quality. In our paper we instead found that offline prediction errors do correlate with online driving quality metrics, but only if an accurate scenario matching technique is adopted, capable of computing the pre-failure site, similar-looking images, and accurate labels.

We have extensively discussed the work by Haq et al. [24], of which this study is a replication. The authors have extended the original paper in a journal version [25] in which they consider one more SDC model to the study (Komanda) and further correlation analysis. However, the threats to validity identified in our study were not addressed. Thus, our results and findings also hold for the extended version of the paper.

### 6.2 Model and System Testing Approaches

Most approaches to testing DNNs that perform autonomous driving are at the model level [20,43,59,72]. For example, DeepXplore [43] uses white-box testing to synthesize inputs that maximize both neuron coverage and behavior diversity. Kim et al. [34] propose several white-box test adequacy criteria based on surprise, defined as the distance in DNN's behaviour between a new, candidate test input and the training data. Inputs should be generated to cover all ranges of surprise, from low to high surprise. DeepTest [59] uses affine transformations from computer vision to produce new inputs that cause the DNN to misbehave. DeepRoad [59] proposes the usage of GANs to generate more realistic driving inputs from streams of real-world data. ThirdEye [50] uses the attention maps from the explainable AI domain to predict misbehaviours of self-driving cars. Dang et al. [20] study the robustness of DNN driving models with respect to different adversarial attacks. Kong et al. [36] generate realistic adversarial billboards within real-world images that are able to confound the vehicle. In our work, we also use universal adversarial perturbations at the system-level, finding comparable results in terms of virtual/physical robustness. However, the focus of our study is on the model vs system level testing comparison.

Concerning *system-level testing* techniques for SDCs, researchers proposed techniques to generate driving scenarios procedurally [3,8,9,41,46,54]. For instance, SilGAN [42] uses GANs to generate driving maneuvers for software-in-the-loop testing. Mullins et al. [41] use Gaussian processes to drive the search towards yet unexplored regions of the input space. Abdessalem et al. [3,8,9] combine genetic algorithms and machine learning to test a pedestrian detection system. Li et al. [37] use ontologies for automatically generating combinatorial test suites for testing automated driving functions. Riccio and Tonella [46] propose a model-based test generator that uses Catmull-Rom splines to characterize the road shape and generate inputs that are at the behavioural frontier of a SDC model. Arri-

eta et al. [5] use a genetic algorithm to generate tests for cyber-physical systems that optimize requirements coverage, test case (dis-)similarity and test execution time. Riccio et al. [44] use mutation adequacy-guided test generation to augment existing test suites for SDCs.

In contrast, our work focuses on the comparison of model vs system-level testing of SDCs, both on a simulated and a real-world environment. Our extension to a physical SDC constitutes a novel contribution to the state of the art.

### 6.3 Challenges for Autonomous Driving Testing

Wotawa [69] discuss the challenges in testing autonomous driving systems and highlight the similarities and the differences with testing safety critical systems. Stellet et al. [49] discuss the testing of advanced driver assistance towards automated driving reporting as main drawback the high initial effort to build the simulation environment, but also the quantification of the achieved degree of realism of such platforms. Riccio et al. [45] present a systematic mapping of the main challenges of testing machine learning-based systems, including autonomous driving systems. A recent work by Zhang et al. [74] provide a comprehensive taxonomy for critical scenario identification methods based on an analysis of the state-of-the-art research, and identify open issues and directions for further research. Wotawa et al. [70] discuss verification and validation methodologies for advanced driver-assistance systems.

Concerning the oracle problem, Kalra et al. [33] calculate the number of miles of driving that would be needed to provide clear statistical evidence of autonomous vehicle safety. Jahangirova et al. [31] evaluated 26 metrics related to the quality of driving of both human and autonomous driving and showed their usefulness as functional oracles through mutation testing [29]. Evans et al. [23] design a domain specific language to express oracles for autonomous driving systems testing such as safety, liveness, timeliness and temporal properties.

Our work compares model- and system-level based testing both in simulated and physical environments, and discusses the conditions under which model- and system-level based testing expose failures.

### 6.4 Physical Testing of Autonomous Vehicles

The usage of physical RC vehicles has fostered substantial research in the domain of autonomous racing, in which DNN malfunctions or deficiencies can have far-reaching safety consequences [10]. Verma et al. [62] compare different scaled vehicles concluding that such platforms allow the rapid exploration of many different test tracks while retaining realistic environmental conditions, which provides further justification for our choice to use Donkey Car. Researchers have been using Donkey Car [39,51,63,73,75] to study also reinforcement learning algorithms for autonomous driving [7,12,35]. Sinha et al. [47] present a framework to predict the vehicle's future state with by experiments on small scale autonomous platforms. Mahmoud et al. [39] use image scaling for functional test of DNN SDC on the Donkey Car platform. Chen et al. [17] embed a real hardware control unit within a simulation platform to verify the validity of self-driving DNNs in virtual

scenes, including perception, planning, decision making, and control. Sotiropoulos et al. [48] report on an exploratory study of bugs in outdoor robots navigation, showing how most of them can be revealed in low-fidelity simulation. Stocco et al. [51] compare virtual and physical testing of autonomous driving systems, reporting a 60% transferability between the two. El Mostadi et al. [40] discuss the drawbacks of virtual testing of advanced driver-assistance systems, including simulation crashes, ill-controlled test executions, incorrect verdict assignments, and waste of time in the running and analysis of useless tests.

Differently from described works, our comparison of model vs system-level testing of SDCs using a real-world physical environment is a novel contribution to the studies using physical platforms.

### 6.5 GAN-based Testing of Autonomous Vehicles

The main focus of existing GAN-based testing techniques is to inject perturbations into a driving scene (e.g., to create realistic weather transition for the same image) for offline testing [72], or to estimate telemetry data that are unavailable in the field, when driving a physical car [42, 71, 51]. DeepRoad [72] uses UNIT [38] to generate accurate photo-realistic paired driving scenes for SDC testing, which were evaluated for their capability of exposing individual prediction errors. SilGAN [42] uses GANs to generate driving maneuvers for software-in-the-loop testing. SurfelGAN [71] is a technique developed at Waymo to generate realistic sensor data for autonomous driving simulation without requiring manual creation of virtual environments and objects. Differently from existing works, we use CycleGAN, that requires no pairing, to generate pseudo-real driving scenes to evaluate the prediction differences between virtual and physical SDCs.

## 7 Conclusions

This paper replicates an existing study on the comparison between model (offline) and system (online) level testing of autonomous driving systems, with a focus on supervised models for lane-keeping. We discussed the main threats to the validity of the original study, and we set up an experimental design that addresses them. Moreover, our study extends the original study, which was conducted only in simulation, with the physical dimension, in which we consider a real-world small-scale self-driving vehicle.

Our experiments did not lead to a full replication of the original study. On the one hand, our study confirms the original findings reported in the replicated study about the possibility to use simulator-generated data as an alternative to real-world data. We obtain a comparably low difference between simulator and real-world prediction errors, and statistical analysis confirm that the distributions of such prediction errors have negligible differences. The latter result does not hold for the data collected in the original study, possibly because of the poor visual similarity affecting the images matched by the heuristic used by the original authors. In our work, matches obtained by automated neural translation result in faithful images across domains (simulator vs real-world).

On the other hand, our study does not confirm the original findings concerning the disagreement between offline and online testing. The observed disagreements consisted of false negatives, i.e., low offline errors associated with system failures. In our replication, such a phenomenon was observed quite rarely and was not as prevalent as in the original study, the main reason being the improved accuracy of our scenario matching technique, which determines precisely the pre-failure sequence, the pairs of online/offline images to match, and the accuracy of the ground truth assigned to each matching pair.

Our results provide strong justification for the ongoing research on simulation-based testing and offline model-level testing since they are both highly correlated with the exposure of real-world failures. Of course, this does not mean that online, in-field testing should be neglected, as it is impossible to account for the variability of the real world in a simulator, or offline. However, by investing in the early levels of testing, companies can reduce substantially the risk of revealing potential deployment failures when approaching a new release of self-driving car software.

## 8 Declarations

### 8.1 Funding and/or Conflicts of interests/Competing interests

This work was partially supported by the H2020 project PRECRIME, funded under the ERC Advanced Grant 2017 Program (ERC Grant Agreement n. 787703). The authors declared that they have no conflict of interest.

### 8.2 Data Availability

Our data, results, and the Donkey Car simulator are available [1].

## References

1. Code artifacts. https://github.com/tsigalko18/emse22 (2022)
2. Many cars have a hundred million lines of code. https://www.technologyreview.com/2012/12/03/181350/many-cars-have-a-hundred-million-lines-of-code/ (2012)
3. Abdessalem, R.B., Panichella, A., Nejati, S., Briand, L.C., Stifter, T.: Testing autonomous cars for feature interaction failures using many-objective search. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, pp. 143–154. ACM, New York, NY, USA (2018). DOI 10.1145/3238147.3238192. URL http://doi.acm.org/10.1145/3238147.3238192
4. Afzal, A., Katz, D.S., Le Goues, C., Timperley, C.S.: Simulation for robotics test automation: Developer perspectives. In: International Conference on Software Testing, Validation and Verification, ICST '21 (2021)
5. Arrieta, A., Wang, S., Markiegi, U., Sagardui, G., Etxeberria, L.: Search-based test case generation for cyber-physical systems. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 688–697 (2017). DOI 10.1109/CEC.2017.7969377
6. AWS Deepracer. https://aws.amazon.com/deepracer (2021)
7. Balaji, B., Mallya, S., Genc, S., Gupta, S., Dirac, L., Khare, V., Roy, G., Sun, T., Tao, Y., Townsend, B., Calleja, E., Muralidhara, S., Karuppasamy, D.: Deepracer: Educational autonomous racing platform for experimentation with sim2real reinforcement learning. CoRR **abs/1911.01562** (2019). URL http://arxiv.org/abs/1911.01562

8. Ben Abdessalem, R., Nejati, S., Briand, L.C., Stifter, T.: Testing advanced driver assistance systems using multi-objective search and neural networks. In: 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 63–74 (2016)
9. Ben Abdessalem, R., Nejati, S., C. Briand, L., Stifter, T.: Testing vision-based control systems using learnable evolutionary algorithms. In: 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), pp. 1016–1026 (2018). DOI 10.1145/3180155.3180160
10. Betz, J., Zheng, H., Liniger, A., Rosolia, U., Karle, P., Behl, M., Krovi, V., Mangharam, R.: Autonomous vehicles on the edge: A survey on autonomous vehicle racing (2022). DOI 10.48550/ARXIV.2202.07008. URL https://arxiv.org/abs/2202.07008
11. BGR Media, LLC: Waymo's self-driving cars hit 10 million miles. https://techcrunch.com/2018/10/10/waymos-self-driving-cars-hit-10-million-miles (2018). Online; accessed 1 September 2021
12. Biagiola, M., Tonella, P.: Testing the plasticity of reinforcement learning based systems. ACM Trans. Softw. Eng. Methodol. (2022). DOI 10.1145/3511701. URL https://doi.org/10.1145/3511701
13. Bojarski, M., Testa, D.D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K.: End to end learning for self-driving cars. CoRR **abs/1604.07316** (2016). URL http://arxiv.org/abs/1604.07316
14. Bulsara, A., Raman, A., Kamarajugadda, S., Schmid, M., Krovi, V.N.: Obstacle avoidance using model predictive control: An implementation and validation study using scaled vehicles. Tech. rep., SAE Technical Paper (2020)
15. Burch, Najm, Yang, Trick: Mcpower: a monte carlo approach to power estimation. In: 1992 IEEE/ACM International Conference on Computer-Aided Design, pp. 90–97 (1992). DOI 10.1109/ICCAD.1992.279392
16. Cerf, V.G.: A comprehensive self-driving car test. Commun. ACM **61**(2), 7–7 (2018). DOI 10.1145/3177753. URL http://doi.acm.org/10.1145/3177753
17. Chen, S., Chen, Y., Zhang, S., Zheng, N.: A novel integrated simulation and testing platform for self-driving cars with hardware in the loop. IEEE Transactions on Intelligent Vehicles **4**(3), 425–436 (2019). DOI 10.1109/TIV.2019.2919470
18. Codevilla, F., López, A.M., Koltun, V., Dosovitskiy, A.: On offline evaluation of vision-based driving models. CoRR **abs/1809.04843** (2018). URL http://arxiv.org/abs/1809.04843
19. Cohen, J.: Statistical power analysis for the behavioral sciences. L. Erlbaum Associates, Hillsdale, N.J (1988)
20. Deng, Y., Zheng, X., Zhang, T., Chen, C., Lou, G., Kim, M.: An analysis of adversarial attacks and defenses on autonomous driving models (2020)
21. Donkey Car. https://www.donkeycar.com/ (2021)
22. Dosovitskiy, A., Ros, G., Codevilla, F., López, A., Koltun, V.: CARLA: an open urban driving simulator. CoRR **abs/1711.03938** (2017). URL http://arxiv.org/abs/1711.03938
23. Evans, A.N., Soffa, M.L., Elbaum, S.: A language for autonomous vehicles testing oracles (2020). URL https://arxiv.org/pdf/2006.10177.pdf
24. Haq, F.U., Shin, D., Nejati, S., Briand, L.: Comparing offline and online testing of deep neural networks: An autonomous car case study. In: Proceedings of 13th IEEE International Conference on Software Testing, Verification and Validation, ICST '20. IEEE (2020)
25. Haq, F.U., Shin, D., Nejati, S., Briand, L.: Can offline testing of deep neural networks replace their online testing? a case study of automated driving systems. Empirical Softw. Engg. **26**(5) (2021). DOI 10.1007/s10664-021-09982-4. URL https://doi.org/10.1007/s10664-021-09982-4
26. Hendrycks, D., Dietterich, T.G.: Benchmarking neural network robustness to common corruptions and perturbations. CoRR **abs/1903.12261** (2019). URL http://arxiv.org/abs/1903.12261
27. How Software Is Eating the Car. https://spectrum.ieee.org/software-eating-car (2021)
28. Humbatova, N., Jahangirova, G., Bavota, G., Riccio, V., Stocco, A., Tonella, P.: Taxonomy of real faults in deep learning systems. ICSE'20. ACM, New York, NY, USA (2020). DOI 10.1145/3377811.3380395

29. Humbatova, N., Jahangirova, G., Tonella, P.: Deepcrime: Mutation testing of deep learning systems based on real faults. In: Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA '21 (2021). DOI 10.1145/3460319.3464825
30. ISO: Road vehicles – Functional safety (2011)
31. Jahangirova, G., Stocco, A., Tonella, P.: Quality metrics and oracles for autonomous vehicles testing. In: Proceedings of 14th IEEE International Conference on Software Testing, Verification and Validation, ICST '21. IEEE (2021)
32. Jain, A., Chaudhari, P., Morari, M.: Bayesrace: Learning to race autonomously using prior experience. CoRR **abs/2005.04755** (2020). URL https://arxiv.org/abs/2005.04755
33. Kalra, N., Paddock, S.M.: Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Transportation Research Part A: Policy and Practice **94**, 182–193 (2016). DOI https://doi.org/10.1016/j.tra.2016.09.010. URL https://www.sciencedirect.com/science/article/pii/S0965856416302129
34. Kim, J., Feldt, R., Yoo, S.: Guiding deep learning system testing using surprise adequacy. In: Proceedings of the 41st International Conference on Software Engineering, ICSE '19, pp. 1039–1049. IEEE Press, Piscataway, NJ, USA (2019). DOI 10.1109/ICSE.2019.00108. URL https://doi.org/10.1109/ICSE.2019.00108
35. Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A.A.A., Yogamani, S., Pérez, P.: Deep reinforcement learning for autonomous driving: A survey (2021)
36. Kong, Z., Liu, C.: Generating adversarial fragments with adversarial networks for physical-world implementation. CoRR **abs/1907.04449** (2019). URL http://arxiv.org/abs/1907.04449
37. Li, Y., Tao, J., Wotawa, F.: Ontology-based test generation for automated and autonomous driving functions. Information and Software Technology **117**, 106,200 (2020). DOI https://doi.org/10.1016/j.infsof.2019.106200. URL https://www.sciencedirect.com/science/article/pii/S0950584918302271
38. Liu, M., Breuel, T.M., Kautz, J.: Unsupervised image-to-image translation networks. CoRR **abs/1703.00848** (2017). URL http://arxiv.org/abs/1703.00848
39. Mahmoud, Y., Okuyama, Y., Fukuchi, T., Kosuke, T., Ando, I.: Optimizing deep-neural-network-driven autonomous race car using image scaling. In: SHS web of conferences, vol. 77, p. 04002. EDP Sciences (2020)
40. Mostadi, M.E., Waeselynck, H., Gabriel, J.M.: Seven technical issues that may ruin your virtual tests for adas. In: 2021 IEEE Intelligent Vehicles Symposium (IV), pp. 16–21 (2021). DOI 10.1109/IV48863.2021.9575953
41. Mullins, G.E., Stankiewicz, P.G., Hawthorne, R.C., Gupta, S.K.: Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles. Journal of Systems and Software **137**, 197–215 (2018). DOI https://doi.org/10.1016/j.jss.2017.10.031. URL http://www.sciencedirect.com/science/article/pii/S0164121217302546
42. Parthasarathy, D., Johansson, A.: Silgan: Generating driving maneuvers for scenario-based software-in-the-loop testing. CoRR **abs/2107.07364** (2021). URL https://arxiv.org/abs/2107.07364
43. Pei, K., Cao, Y., Yang, J., Jana, S.: Deepxplore: Automated whitebox testing of deep learning systems. In: Proceedings of the 26th Symposium on Operating Systems Principles, SOSP '17, pp. 1–18. ACM, New York, NY, USA (2017). DOI 10.1145/3132747.3132785. URL http://doi.acm.org/10.1145/3132747.3132785
44. Riccio, V., Humbatova, N., Jahangirova, G., Tonella, P.: DeepMetis: Augmenting a deep learning test set to increase its mutation score. In: Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering, ASE '21. IEEE/ACM (2021)
45. Riccio, V., Jahangirova, G., Stocco, A., Humbatova, N., Weiss, M., Tonella, P.: Testing Machine Learning based Systems: A Systematic Mapping. Empirical Software Engineering (2020)
46. Riccio, V., Tonella, P.: Model-Based Exploration of the Frontier of Behaviours for Deep Learning System Testing. In: Proceedings of ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE '20 (2020)
47. Sinha, A., O'Kelly, M., Zheng, H., Mangharam, R., Duchi, J., Tedrake, R.: Formulazero: Distributionally robust online adaptation via offline population synthesis (2020). DOI 10.48550/ARXIV.2003.03900. URL https://arxiv.org/abs/2003.03900
48. Sotiropoulos, T., Waeselynck, H., Guiochet, J., Ingrand, F.: Can robot navigation bugs be found in simulation? an exploratory study. In: 2017 IEEE International Conference on

Software Quality, Reliability and Security (QRS), pp. 150–159 (2017). DOI 10.1109/QRS.2017.25
49. Stellet, J.E., Zofka, M.R., Schumacher, J., Schamm, T., Niewels, F., Zöllner, J.M.: Testing of advanced driver assistance towards automated driving: A survey and taxonomy on existing approaches and open questions. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pp. 1455–1462 (2015). DOI 10.1109/ITSC.2015.236
50. Stocco, A., Nunes, P.J., d'Amorim, M., Tonella, P.: ThirdEye: Attention maps for safe autonomous driving systems. In: Proceedings of 37th IEEE/ACM International Conference on Automated Software Engineering, ASE '22. IEEE/ACM (2022)
51. Stocco, A., Pulfer, B., Tonella, P.: Mind the Gap! A Study on the Transferability of Virtual vs Physical-world Testing of Autonomous Driving Systems. IEEE Transactions on Software Engineering (2022). URL https://arxiv.org/abs/2112.11255
52. Stocco, A., Tonella, P.: Towards anomaly detectors that learn continuously. In: Proceedings of 31st International Symposium on Software Reliability Engineering Workshops, ISSREW 2020. IEEE (2020)
53. Stocco, A., Tonella, P.: Confidence-driven weighted retraining for predicting safety-critical failures in autonomous driving systems. Journal of Software: Evolution and Process (2021). DOI 10.1002/smr.2386. URL https://doi.org/10.1002/smr.2386
54. Stocco, A., Weiss, M., Calzana, M., Tonella, P.: Misbehaviour prediction for autonomous driving systems. In: Proceedings of 42nd International Conference on Software Engineering, ICSE '20. ACM (2020)
55. Tawn Kramer, M.E., contributors: Donkeycar. https://www.donkeycar.com/ (2022)
56. Team Autumn: Steering angle model: Autumn. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/autumn (2016). Online; accessed 1 September 2021
57. Team Chauffeur: Steering angle model: Chauffeur. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/chauffeur (2016). Online; accessed 1 September 2021
58. Thorn, E., Kimmel, S.C., Chaka, M.: A framework for automated driving system testable cases and scenarios (2018)
59. Tian, Y., Pei, K., Jana, S., Ray, B.: Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In: Proceedings of the 40th International Conference on Software Engineering, ICSE '18, pp. 303–314. ACM, New York, NY, USA (2018). DOI 10.1145/3180155.3180220. URL http://doi.acm.org/10.1145/3180155.3180220
60. Udacity self-driving challenge 2, ch2-001 (testing) and ch2-002 (training). https://github.com/udacity/self-driving-car/tree/master/datasets/CH2 (2016)
61. Unity3d. https://unity.com (2019)
62. Verma, A., Bagkar, S., Allam, N.V.S., Raman, A., Schmid, M., Krovi, V.N.: Implementation and validation of behavior cloning using scaled vehicles. In: SAE WCX Digital Summit. SAE International (2021). DOI https://doi.org/10.4271/2021-01-0248. URL https://doi.org/10.4271/2021-01-0248
63. Viitala, A., Boney, R., Kannala, J.: Learning to drive small scale cars from scratch. CoRR **abs/2008.00715** (2020). URL https://arxiv.org/abs/2008.00715
64. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612 (2004). DOI 10.1109/TIP.2003.819861
65. Waymo Driver. https://waymo.com/waymo-driver/ (2021)
66. Waymo LLC: Waymo Open Dataset. https://waymo.com/open/ (2021). Online; accessed 1 September 2021
67. Waymo Secret Testing. https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/ (2017)
68. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6), 80 (1945). DOI 10.2307/3001968. URL https://doi.org/10.2307/3001968
69. Wotawa, F.: Testing Autonomous and Highly Configurable Systems: Challenges and Feasible Solutions, pp. 519–532. Springer International Publishing, Cham (2017). DOI 10.1007/978-3-319-31895-0_22. URL https://doi.org/10.1007/978-3-319-31895-0_22
70. Wotawa, F., Klück, F., Zimmermann, M., Nica, M., Felbinger, H., Tao, J., Li, Y.: Recent Verification and Validation Methodologies for Advanced Driver-Assistance Systems. CRC Press (2021)
71. Yang, Z., Chai, Y., Anguelov, D., Zhou, Y., Sun, P., Erhan, D., Rafferty, S., Kretzschmar, H.: Surfelgan: Synthesizing realistic sensor data for autonomous driving. CoRR **abs/2005.03844** (2020). URL https://arxiv.org/abs/2005.03844

72. Zhang, M., Zhang, Y., Zhang, L., Liu, C., Khurshid, S.: Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, pp. 132–142. ACM, New York, NY, USA (2018). DOI 10.1145/3238147.3238187. URL http://doi.acm.org/10.1145/3238147.3238187
73. Zhang, Q., Du, T.: Self-driving scale car trained by deep reinforcement learning. CoRR **abs/1909.03467** (2019). URL http://arxiv.org/abs/1909.03467
74. Zhang, X., Tao, J., Tan, K., Torngren, M., Gaspar Sanchez, J.M., Ramli, M.R., Tao, X., Gyllenhammar, M., Wotawa, F., Mohan, N., Nica, M., Felbinger, H.: Finding critical scenarios for automated driving systems: A systematic mapping study. IEEE Transactions on Software Engineering pp. 1–1 (2022). DOI 10.1109/TSE.2022.3170122
75. Zhou, H., Chen, X., Zhang, G., Zhou, W.: Deep reinforcement learning for autonomous driving by transferring visual features. In: 2020 25th International Conference on Pattern Recognition (ICPR), pp. 4436–4441 (2021). DOI 10.1109/ICPR48806.2021.9412011
76. Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. CoRR **abs/1703.10593** (2017). URL http://arxiv.org/abs/1703.10593